Computer Science and Engineering
IIIT Kalyani, West Bengal

Compilers Design Laboratory (CS 511)
(Autumn: 2019 - 2020)

3rd Year CSE: 5th Semester

Assignment - 6 *Marks: 10*

Assignment Out: *11th October, 2019* Report on or before: *18th October, 2019*

1. Consider the following *context-free grammar* G: the *non-terminals* are $N = \{$ *AS, BE, D, DL, E, IOS, IS, LS, P, PE, RE, S, SL, TY, VL* $\}$,
the *terminals* are
$\Sigma = \{ + - * / == < > ( ) = ;$ **& | ~ else ic fc id if int real do nop print read str then while** $\}$,
the *start symbol* is P and the production rules are,

$$
\begin{array}{rcl}
P & \rightarrow & DL\ SL \\
DL & \rightarrow & DL\ D \mid \varepsilon \\
D & \rightarrow & VL\ \mathbf{:}\ TY \\
TY & \rightarrow & \mathbf{int} \mid \mathbf{real} \\
VL & \rightarrow & VL\ \mathbf{id} \mid \mathbf{id} \\
SL & \rightarrow & SL \mid S \\
S & \rightarrow & AS \mid IS \mid LS \mid IOS \mid \mathbf{nop} \\
AS & \rightarrow & \mathbf{id} = E \\
IS & \rightarrow & \mathbf{if}\ BE\ \mathbf{then}\ SL\ \mathbf{else}\ SL\ \mathbf{;} \\
LS & \rightarrow & \mathbf{while}\ BE\ \mathbf{do}\ SL\ \mathbf{;} \\
IOS & \rightarrow & \mathbf{print}\ PE \mid \mathbf{read\ id} \\
PE & \rightarrow & E \mid \mathbf{str} \\
BE & \rightarrow & BE\ \text{`}\mid\text{'}\ BE \mid BE\ \mathbf{\&}\ BE \mid \mathbf{\sim}\ BE \mid (\ BE\ ) \mid RE \\
RE & \rightarrow & E == E \mid E < E \mid E > E \\
E & \rightarrow & E + E \mid E - E \mid E * E \mid E\ /\ E \mid - E \mid (\ E\ ) \mid \mathbf{id} \mid \mathbf{ic} \mid \mathbf{fc}
\end{array}
$$

Most of the terminals have their usual meaning e.g. **id** is an *identifier*, **ic** is an integer constant, **fc** is a floating-point constant, **str** is a string of characters, **nop** for *no operation*, = is an assignment, == is equality, '|' is *logical 'or'*, **&** is *logical 'and'*, **~** is '*not*' etc. Both '|' and '&' are *left associative*. The precedence relations of logical operators are '|' < '&' < '~'.

An *identifier* starts with an English alphabet followed by a sequence of alphabet or decimal digits. An integer constant is a sequence of decimal digits. a floating-point constant is a sequence of decimal digits with a *decimal dot* in it $(.12, 12., 1.2)$.

A comment starts with '//' and is up to the end of line ('\ n'). A character *string* is within a pair of quotes (").

2. Write a *flex* specification of the scanner. The name of the *flex* specification file should be `<roll no>.l`. The command
`$ flex <roll no>.l`
generate the C code for the scanner in `lex.yy.c`. Include the header file `<roll no>.tab.h++` in the definition section of your *flex* specification. This header file, created by *bison*, contains the token names, type of `yylval` etc.

3. Write *bison* specification for the grammar to generate a parser. You need to specify the precedence and associativity of *boolean* and *arithmetic* operators as these parts of the production rules makes the grammar *ambiguous*. The specification file name should be `<roll no>.y++` (this will generate C++ parser code). The command
`$ bison -d -v <roll no>.y++`
generates three files
`<roll no>.tab.c++`,
`<roll no>.tab.h++`, and

`<roll no>.output`.
The `.output` file contains the description of the LALR(1) automaton.

4. The compiled code of `<roll no>.tab.c` and `lex.yy.c` gives the basic parser. Let the name of the executable file be `myParser`. Prepare a Makefile for the whole process.

5. To send the assignment for evaluation prepare a `tar` file with the name `<roll no>.6.tar` which includes three (3) or more files: `Makfile`, `<roll no>.l`, `<roll no>.y++`. Kindly do not put them under a subdirectory while preparing the `tar` file.

6. A sample input is

```
// This program computes factorial
n fact i : int
read n
i = 1
fact = 1
while i < n | i == n do
    fact = fact * i
    i = i + 1
;   // Note the ;
print fact
```

7. You may print the syntax error message by defining `yylineno` in *flex* (`%option yylineno`) and using it with the `yyerror()` in *bison*. As an example -

```
// This program computes factorial
n fact i : int
read n
i = 1
fact = 1
while i  n | i == n do
    fact = fact * i
    i = i + 1
;   // Note the ;
print fact

syntax error at line no: 6
```